PQL: KONVERSI SINTAKS SQL MENJADI SINTAKS PQL

Benhard Sitohang, Tricya Esterina Widagdo
Data & Software Engineering Research Division, School of Electrical Engineering and Informatics,
Institut Teknologi Bandung
E-mail: benhard@stei.itb.ac.id, cia@informatika.org

ABSTRAK: Evaluasi cakupan semantik PQL (Public Query Language) sebagai satu query, dibandingkan dengan SQL, merupakan kajian utama pada penelitian tentang konversi yang dijelaskan pada makalah ini. Pendekatan dalam evaluasi tersebut, dilakukan dengan mengembangkan dan uji-coba konversi dari SQL menjadi PQL, dengan menggunakan operator relasi sebagai media konversi. Dari pengembangan konversi ini telah dapat mengelompokkan, bahwa semua sintaks SQL yang didasarkan pada natural join dan melibatkan 1 atau beberapa relasi, dapat dikonversi menjadi sintaks PQL, baik query yang bersarang (nested) maupun tidak bersarang.

Kata kunci: *Public Query Language (PQL), Structured Query Language (SQL),* model relasi, operator relasi**.**

***ABSTRACT**: The aim of our research presented was to evaluate the semantic coverability of PQL (Public Query Language) as a query language, compared to SQL (Structured Query Language). Our approach in doing the evaluation was to develop conversion rules from SQL to PQL statements, by using relational expressions as the intermediate language. The SQL syntaxes were grouped into four categories, based on the number of relations used in the queries and whether the queries were nested or not. Based on the conversions results, it can be concluded that all of SQL statements that include one or more relations with natural join operations – with or without nested sub queries – can be converted into PQL statements.*

***Keywords**: Public Query Language (PQL), Structured Query Language (SQL), relational model, relational operator.*

PENDAHULUAN

Kemudahan pemahaman sintaks PQL dibandingkan SQL terutama sebagai dampak dari pengurangan klausa FROM, yang selanjutnya berdampak pada berkurangnya tuntutan pemahaman yang diperlukan oleh pemakai, sehingga dapat membentuk query berdasarkan sintaks PQL [3, 6].

Selanjutnya, pengurangan klausa FROM juga berdampak penyederhanaan pada klausa WHERE (hanya untuk kondisi sederhana, tidak memungkinkan kondisi rekursif), yang pada akhirnya membatasi cakupan fungsi yang dapat dijabarkan oleh pemakai dalam bentuk query PQL.

Untuk mengetahui sejauh mana cakupan fungsi yang dapat dijabarkan dengan menggunakan sintaks PQL, telah dilakukan penelitian sebagaimana yang dijabarkan dalam makalah ini, yang dimaksudkan untuk mengevaluasi cakupan fungsi yang ada pada PQL, relatif dibandingkan dengan cakupan fungsi pada sintaks SQL.

Evaluasi yang dimaksudkan melalui cara konversi pada makalah ini, dibatasi hanya untuk sintaks SQL untuk retrieve data (baik bersarang/nested dengan satu relasi atau lebih, maupun sintaks tidak bersarang dengan melibatkan satu relasi atau lebih pada klausa FROM, menjadi sintaks PQL. Algortima rinci untuk konversi SQL menjadi PQL, dijelaskan pada [9].

Beberapa penelitan dalam lingkup PQL telah dilakukan untuk beberapa aspek sejak deklarasi sintaks lengkap PQL pada tahun 1995 [3], baik dalam arti algoritma untuk eksekusi PQL, yang lebih dikenal dengan jaringan semantik data [5], konversi PQL menjadi SQL agar dapat dieksekusi oleh prosessor SQL [4, 10].

Ide penyederhanaan yang dimaksudkan pada PQL, relatif sama dengan apa yang dimaksudkan pada *Microsoft English Query (MEL)*, tetapi dengan pendekatan yang berbeda. Prinsip utama pada MEL: penyediaan penterjemah antara bahasa alami (bahasa Inggris) dengan SQL, dengan menyediakan berbagai term spesifik pada skema basis data sesuai dengan kebiasaan pemakai memandang skema termaksud [2]. Dengan demikian, cakupan MEL dibatasi oleh term terjemahan yang sudah ada dalam sistem. Sedangkan pada PQL, tidak dilakukan dengan prinsip penterjemahan, melainkan sistem PQL menggunakan skema virtual, yaitu berupa flat file (universal relation) [8]. Dengan demikian, pemakai akan memandang basis data seperti halnya hanya terdiri dari 1 file.

STRUCTURED QUERY LANGUAGE (SQL)

SQL merupakan bahasa penelusuran pada basis data relasional yang didasarkan pada satu sintaks select-from-where khusus untuk *retrieve* data. Sintaks Query SQL juga memberikan fasilitas untuk query bersarang, yaitu ekspresi query bersarang pada query lain yang digunakan untuk menguji keanggotaan dan perbandingan.

Bentuk umum SQL linier adalah sebagai berikut [7]: ***SELECT*** [***DISTINCT***] <list atribut> [***INTO*** <nama-relasi >] ***FROM*** {<nama-relasi> | <nama-view>} $^{n}_{1}$ [***WHERE*** <eksp>] [***GROUP BY*** [***ALL***] <atribut>, <atribut>] … [***HAVING*** <eksp>] [***ORDER BY*** [{<nama-relasi> | <nama-view>}] <atribut> | <list-nomor> | <eksp> [***ASC*** | ***DESC***], [{<nama-relasi> | <nama-view>}] <atribut> | <list-nomor> | <eksp> [***ASC*** | ***DESC***], …]

Adapun bentuk umum Sintaks Query SQL bersarang adalah adanya query seperti bentuk linier, di dalam klausa where [7].

PUBLIC QUERY LANGUAGE (PQL)

PQL adalah sintaks query yang dibangun dengan sintaks SELECT–WHERE [3], yang dimaksudkan untuk memberi kemudahan bagi pemakai awam (relatif, dibandingkan dengan SQL), dengan cara menghilangkan kata kunci FROM. Dengan menghilangkan kata kunci tersebut, secara prinsip ada beberapa hal yang tidak perlu dipahami oleh pemakai, yaitu : Relasi, Atribut kunci, Operasi dalam kelompok Join, dan Ketergantungan fungsional.

Sintaks PQL dibangun dari 2 dan hanya 2 klausa, yaitu: **TAMPILKAN / SELECT {<atribut>}$^{n}_{1}$ {JIKA / WHERE <eksp>}$^{1}_{0}$;**

Sintaks lengkap PQL dijabarkan pada tabel 1.

Tabel 1. Sintaks PQL

SINTAKS PQL

SELECT {<attribute>} $^{n}1$ {WHERE < eksp>} ;
<eksp> : : = <eksp1>|(< eksp1>) {<oplojbin> <eksp> }$^{n}0$ |
NOT (<eksp1>) { < oplojbin > <eksp>}$^{n}0$;
<eksp1> : : = < eksp2 > |<eksp2> < oplojbin> < eksp> } ;
<eksp2> : : = < attribute> < oprel> {<attribute>
<konstanta> } ;
<oplojbin > : : = **DAN** | **ATAU** ;
<oprel> : : = = | < > | < | > | < = | < = ;
<atribut> : : = {<huruf > <huruf>|<angka>|} $^{n}0$;
<konstanta> : : = < string > | < bilangan> ;
<string > : : = {<huruf>|<angka>|<spesial>} $^{n}0$;
<bilangan> : : = {|+}$^{1}0${<bulat >| <bulat> }$^{1}0$;
<bulat > : : = { < angka > } $^{n}1$;
<angka> : : = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |8 | 9 ;
<huruf > : : = a | b| c | d| ... |z | A | B| C |.... Z| ;
<spesial> : : = / | / | * | (|) | ^ | % | $ | # | @ | ! | ?
| . | , | ” | ’ | [|] | { | } | | | ` | ~ | : | ; | + | − | _ ;

BATASAN KONVERSI

Konversi yang dimaksudkan dalam makalah ini (konversi SQL menjadi PQL) dibatasi pada query untuk *retrieve*, dan tidak mencakup:

a. Sintaks Query SQL dengan DISTINCT,INTO, EXIST, UNIQUE, GROUP BY-HAVING, ALL, ORDER BY-ASC, ORDER BY-DESC dan agregat fungsi.
b. Sintaks SQL dengan GROUP BY. Sintaks dengan GROUP BY secara implisit sudah dilakukan pada hasil tampilan PQL, karena hasil query pada PQL sudah dikelompokkan berdasarkan atribut yang dipilih pertama kali.
c. Sintaks DISTINCT dan UNIQUE. Sintaks tersebut tidak diperlukan, karena PQL memberikan hasil yang tidak terduplikasi.

PENGELOMPOKAN SINTAKS SQL DAN PRINSIP KONVERSI

Untuk medukung konversi, sintaks SQL (meliputi Sintaks Query SQL tidak bersarang dan bersarang baik terdefinisi maupun tidak terdefinisi pada PQL) dikelompokkan berdasarkan jumlah relasi yang dilibatkan, agar terlihat jelas bentuk natural joinnya. Untuk kemudahan penjelasan tentang konversi, sintaks SQL dikelompokkan dalam 4 kategori, yaitu:

- Sintaks Query SQL tidak bersarang, melibatkan 1 relasi
- Sintaks Query SQL tidak bersarang, melibatkan n relasi
- Sintaks Query SQL bersarang, melibatkan 1 relasi
- Sintaks Query SQL bersarang, melibatkan n relasi

Aturan konversi secara umum adalah sbb.:

- Semua atribut yang diproyeksi oleh SQL akan ditampilkan oleh PQL
- Relasi pada SQL tidak akan muncul pada PQL karena PQL tidak membebani penggunanya dengan pengetahuan/pengenalan tentang skema relasi
- Semua ekspresi yang mengikuti klausa where, selama predikat itu bukan untuk mendefinisikan natural join, akan menjadi kondisi PQL.

SQL BERBASIS NATURAL JOIN

Berikut ini akan dijabarkan Sintaks SQL yang terdefinisi pada PQL, yaitu yang berbasiskan natural join.

SQL Tidak Bersarang

1 (Satu) Relasi

Bentuk umum Sintaks Query SQL tidak bersarang melibatkan 1 relasi pada Tabel 2.

Tabel 2. Sintaks Query SQL 1 Relasi

```
Select {<atribut>}^n_1
From <nama-relasi>
[Where <eksp>]
<eksp>::=<eksp1>|(<eksp1>)
{<oplojbin><eksp>}^n_0| not
(<eksp1>){<oplojbin><eksp>}^n_0
```

Penjelasan ekivalensi dengan Sintaks Query PQL: semua atribut yang diproyeksi oleh SQL akan mengikuti klausa Tampilkan pada PQL dan ekspresi yang mengikuti klausa where akan menjadi ekspresi PQL setelah klausa Jika. Pembuktian pada beberapa contoh kasus berikut menggunakan **r** untuk relasi yang terlibat pada query SQL dan **R** untuk relasi yang terlibat pada query PQL.

Berikut contoh kasus Sintaks Query SQL tidak bersarang melibatkan 1 relasi:

Kasus 1 :

Sintaks Query **SQL**:

Select $A_1, A_2,\ldots, A_n$
From r
Where p_1

Sintaks Query **PQL**:

Tampilkan $A_1, A_2,\ldots, A_n$
Jika p_1

Dengan aljabar relasional dibuktikan SQL = PQL, yaitu:

$\prod_{A1,A2,\ldots,An}(\sigma_{p1}(r)) = \prod_{A1,A2,\ldots,An}(\sigma_{p1}(R))$
$= \prod_{A1,A2,\ldots,An}(\sigma_{p1}(r))$

Penjelasan pembuktian di atas adalah sebagai berikut : sintaks Query PQL tidak mengenal relasi, dan relasi yang terlibat dibentuk menjadi suatu relasi datar (virtual flat relation atau Universal Relation) [1, 7, 8], maka R = r karena hanya ada satu relasi.

n Relasi

Bentuk umum Sintaks Query SQL tidak bersarang melibatkan n relasi seperti pada Tabel 3.

Tabel 3. Sintaks Query SQL n Relasi

```
Select {<relasi.atribut>}^n_1
From {<nama-relasi>}^n_2
Where <neksp> ;
<neksp> ::= [<eksp><and>]<join-list-
atribut-sama>
<join-list-atribut-sama>::=
{<relasi.atribut::= relasi.atribut>
[<and>]}^n_1
```

Penjelasan ekivalensi dengan Sintaks Query PQL: pada bentuk Query SQL tidak bersarang melibatkan n relasi, klausa where minimal harus berisi join list atribut semua relasi yang berada pada klausa from untuk membentuk natural join agar dapat dikonversi menjadi Sintaks Query PQL. Kondisi SQL selain ekspresi untuk membentuk natural join akan menjadi ekspresi PQL setelah klausa Jika.

Berikut contoh kasus Sintaks Query SQL tidak bersarang melibatkan n relasi.

Kasus 2:

Misal terdapat relasi:

r $(A_1, A_2,\ldots, A_n)$
s $(A_1, B_1,\ldots, B_m)$

Sintaks Query **SQL**:

Select r.A_1, $A_2,\ldots, A_n$, $B_1,\ldots, B_m$
From r, s
Where A_2 = "cr" and r.A_1 = s.A_1

Sintaks Query **PQL**:

Tampilkan $A_1, A_2,\ldots, A_n$, $B_1,\ldots, B_m$
Jika A_2 = "cr"

Pembuktian bahwa Sintaks Query SQL = Sintaks Query PQL, yaitu: Aljabar relasional Sintaks Query SQL:

$\prod_{A1,A2,\ldots,An,B1,\ldots,Bm}(\sigma_{r.A1=s.A1}(\sigma_{r.A2=cr}(rxs)))$
$= \prod_{A1,A2,\ldots,An,B1,\ldots,Bm}(\sigma_{r.A2=cr \wedge r.A1=s.A1}(rxs)))$
$= \prod_{A1,A2,\ldots,An,B1,\ldots,Bm}(\sigma_{r.A2=cr}(r[X]s))$

Join pada query SQL ini adalah *natural join.*

Aljabar relasional dari Sintaks Query PQL:

$\prod_{A1,A2,\ldots,An,B1,\ldots,Bm}(\sigma_{A2=cr}(R))$

SQL = PQL, yaitu :

$\prod_{A1,A2,\ldots,An}(\sigma_{r.A2=cr}(r[X]\ s))$
$= \prod_{A1,A2,\ldots,An}(\sigma_{A2=cr}(R))$
$= \prod_{A1,A2,\ldots,An}(\sigma_{A2=cr}(r[X]s))$
$= \prod_{A1,A2,\ldots,An}(\sigma_{r.A2=cr}(r[X]\ s))$

Penjelasan pembuktian di atas adalah sebagai berikut: *Virtual flat relation* pada PQL adalah R = r [X] s dan atribut A_2 sebenarnya terdapat para relasi r maka Sintaks Query PQL di atas ekivalen dengan Sintaks Query SQL.

SQL Bersarang

1 (satu) Relasi

Bentuk umum Sintaks Query SQL bersarang melibatkan 1 relasi seperti pada Tabel 4.

Tabel 4. Sintaks Query SQL 1 Relasi

```
Select {<atribut>}n1
From <nama-relasi>
Where <nesteksp> ;
<nesteksp> ::=<nesteksp1>|
(<nesteksp1>)<oplojbin>
(<nesteksp1>);
<nesteksp1>::=[<eksp><and>] <atribut>
<in>(<qeksp>) ;
<qeksp> ::= <select atribut>
<from><nama-relasi> [<where> <eksp>
|<nesteksp>]
```

Penjelasan ekivalensi dengan Sintaks Query PQL: pada SQL bersarang melibatkan 1 relasi, ekspresi yang akan diambil menjadi kondisi pada PQL setelah klausa jika hanya ekspresi yang bukan menyatakan join antar relasi dan pemberian prioritas eksekusi untuk ekspresi bersarang terdalam.

Berikut contoh kasus SQL bersarang melibatkan 1 relasi:

Kasus 3

Misal terdapat relasi:

r ($A_1, A_2, \ldots, A_n$)
s ($A_1, B_1, \ldots, B_m$)

Sintaks Query SQL bersarang adalah sebagai berikut:

Sintaks Query **SQL:**

Select $A_1, A_2, \ldots, A_n$
From r
Where A_i = cr ***and*** A_1 ***in***
  Select A_1
  From s
  Where B_j = cs

Sintaks Query **PQL :**

Tampilkan $A_1, A_2, \ldots, A_n$
Jika A_i = cr ***dan*** B_j = cs

Pembuktian bahwa Sintaks Query SQL = PQL, yaitu: Sintaks Query SQL diatas bila ditulis dalam aljabar relasional adalah (dari query terdalam):

Select A_1
From s
Where B_j = cs

Dari Sintaks Query SQL terdalam ini diperoleh tuple sebanyak k, sehingga terdapat tuple t_1 [h] … t_k [h] dengan k >=0. ljabar relasional dari Sintaks Query SQL ini adalah: $\prod_{A1}(\sigma_{Bj=cs}(s))$

Sedangkan dari query, adalah:

A_1 ***in***
Select A_1 ***From*** s
Where B_j = cs

Join antara relasi **s** dan relasi **r** adalah melalui atribut A_1, sehingga aljabar relasional yang menyatakan join ini adalah:

$$\prod_{A1}(r)[X]\prod_{A1}(\sigma_{Bj=cs}(s))=\prod_{A1}(\sigma_{Bj=cs}(r[X]s))$$

Dari query terluar
Select $A_1, A_2, \ldots, A_n$
From r
Where A_i = cr ***and*** A_1 ***in***
Select A_1 ***From*** s ***Where*** B_j = cs

dapat dinyatakan aljabar relasionalnya sebagai berikut:

$$\prod_{A1,A2,\ldots,An}(\sigma_{Ai=cr \wedge (r.A1=t1[h] \vee \ldots \vee r.A1=tk[h])}(r))$$
$$=\prod_{A1,A2,\ldots,An}(\sigma_{Ai=cr \wedge Bj=cs}(r[X]s))$$

Sintaks Query PQL diatas bila dijelaskan dalam aljabar relasional adalah:

$$\prod_{A1,A2,\ldots,An}(\sigma_{Ai=cr \wedge Bj=cs}(R))$$

Pembuktian SQL = PQL:

$$\prod_{A1,A2,\ldots,An}(\sigma_{Ai=cr \wedge Bj=cs}(r[X]s)))$$
$$=\prod_{A1,A2,\ldots,An}(\sigma_{Ai=cr \wedge Bj=cs}(R))$$
$$=\prod_{A1,A2,\ldots,An}(\sigma_{Ai=cr \wedge Bj=cs}(r[X]s))$$

Penjelasan pembuktian di atas adalah sebagai berikut: *Virtual flat relation* pada Sintaks Query PQL sebenarnya adalah R = r [X] s dengan atribut join adalah A_1, oleh karena itu Sintaks Query PQL diatas ekivalen dengan Sintaks Query SQL.

n Relasi

Bentuk umum Sintaks Query SQL bersarang yang melibatkan n relasi adalah sebagaimana ditampilkan pada Tabel 5.

Tabel 5. Sintaks Query SQL n Relasi

```
Select {<relasi.atribut>}n1
  From {<nama-relasi>}2n
  Where <nesteksp>
<nesteksp>  ::= <nesteksp1> |
(<nesteksp1>)<oplojbin>(<nesteksp1>)
<nesteksp1> ::= [<eksp><and>] <join
list-atribut sama>
<and>[<relasi.>]<atribut><in>(<qeksp>)
<qeksp> ::= <select atribut>
<from><nama-relasi> [<where>
<eksp>|<nesteksp>]
```

Penjelasan ekivalensi dengan Sintaks Query PQL: Pada SQL bersarang melibatkan n relasi, ekspresi yang akan diambil menjadi kondisi pada PQL setelah klausa jika hanya ekspresi yang bukan menyatakan natural join antar relasi dan pemberian prioritas eksekusi untuk ekspresi bersarang terdalam.

Berikut contoh kasus SQL bersarang melibatkan n relasi:

Kasus 4
Misal terdapat relasi:

r ($A_1, A_2, \ldots, A_n$)
s ($A_1, B_1, \ldots, B_m$)
p($A_1, C_1, \ldots, C_o$)

Sintaks Query **SQL**:
Select $r.A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m$
From r, s
Where $r.A_i = cr$ and $s.B_j = cs$ and $r.A_1 = s.A_1$ and $r.A_1$ ***in***
Select A_1 ***From*** p ***Where*** $p.C_k = cp$

Sintaks Query **PQL**:
Tampilkan $A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m$
Jika $A_i = cr$ dan $B_j = cs$ dan $C_k = cp$

Pembuktian bahwa Sintaks Query SQL = PQL, yaitu, Aljabar relasional dari Sintaks Query SQL adalah sebagai berikut:

$\prod_{A1, A2, \ldots, An, B1, B2, \ldots, Bm} (\sigma_{r.A1 = s.A1 \wedge s.Bj = cs \wedge r.Ai = cr} (\sigma_{p.Ck = cp} (r \times s) [X] p))$
$= \prod_{A1,A2,\ldots,An,B1,B2,\ldots,Bm} (\sigma_{s.Bj=cs \wedge r.Ai=cr}$
$(\sigma_{p.Ck=cp} (r[X]s)) [X]p)$
$= \prod_{A1,A2,\ldots,An,B1,B2,\ldots,Bm}$
$(\sigma_{s.Bj=cs \wedge r.Ai=cr \wedge p.Ck=cp} (r[X]s[X]p))$

Aljabar relasional dari Sintaks Query PQL adalah sebagai berikut:

$\prod_{A1, A2, \ldots, An, B1, B2, \ldots, Bm} (\sigma_{Ai = cr \wedge Bj = cs \wedge Ck = cp} (R))$

Pembandingan aljabar relasional untuk membuktikan SQL = PQL, yaitu:

$\prod_{A1,A2,\ldots,An,B1,B2,\ldots,Bm} (\sigma_{s.Bj=cs \wedge r.Ai=cr \wedge p.Ck=cp} (r[X]s[X]p))$
$= \prod_{A1,A2,\ldots,An,B1,B2,\ldots,Bm} (\sigma_{Ai=cr \wedge Bj=cs \wedge Ck=cp} (R))$
$= \prod_{A1,A2,\ldots,An,B1,B2,\ldots,Bm} (\sigma_{Ai=cr \wedge Bj=cs \wedge Ck=cp} (r[X]s[X]p))$
$= \prod_{A1,A2,\ldots,An,B1,B2,\ldots,Bm} (\sigma_{s.Bj=cs \wedge r.Ai=cr \wedge p.Ck=cp}$
$(r[X]s[X]p))$

Penjelasan pembuktian di atas adalah sebagai berikut : *Virtual flat relation* pada Sintaks Query PQL sebenarnya adalah R = r [X] s [X] p dengan atribut join adalah A_1, oleh karena itu Sintaks Query PQL di atas ekivalen dengan Sintaks Query SQL.

SQL TIDAK BERBASIS NATURAL JOIN

Sintaks Query SQL yang tidak berbasiskan natural join tidak dapat dikonversi menjadi Sintaks Query PQL, bentuk-bentuk Sintaks Query SQL tersebut dijelaskan berikut ini.

Cartesian Product

Penggabungan dua relasi atau lebih dengan cartesian product adalah dengan mencari semua kemungkinan pasangan tuple antar relasi tersebut. Relasi yang dihasilkan dengan penggabungan seperti ini tidak terdefinisi pada PQL karena dasar PQL adalah **natural join.**

Kasus 5
Misal terdapat relasi:

r ($A_1, A_2, \ldots, A_n$)
s ($A_1, B_1, \ldots, B_m$)

Sintaks Query **SQL:**
Select $A_1, A_2, \ldots, A_n, B_1, \ldots, B_m$
From r, s
Where $r.A_1 = cr$
Sintaks Query **PQL:** Tidak terdefinisi

Penjelasan Sintaks Query SQL di atas tidak terdefinisi pada PQL adalah sebagai berikut: Aljabar relasional dari Sintaks Query SQL sebagai berikut:
$\prod_{A1,A2,\ldots,An} (\sigma_{r.A1 = cr} (r \times s))$
Join pada query SQL ini adalah *cartesian product.*

Kasus 6
Misal terdapat relasi:

r ($A_1, A_2, \ldots, A_n$)
s ($A_1, B_1, \ldots, B_m$)
p($A_1, C_1, \ldots, C_o$)

Sintaks Query **SQL**:
Select $r.A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m$ ***From*** r,s
Where $r.A_i = cr$ and $s.B_j = cs$ and $r.A_1$ ***in***
Select A_1 ***From*** p ***Where*** $p.C_k = cp$

Sintaks Query **PQL**: Tidak terdefinisi

Penjelasan Sintaks Query SQL di atas yang tidak terdefinisi pada PQL adalah sebagai berikut: Aljabar relasional dari Sintaks Query SQL sebagai berikut:
$\prod_{A1,A2,\ldots,An,B1,B2,\ldots,Bm} (\sigma_{s.Bj = cs \wedge r.Ai = cr} (\sigma_{p.Ck = cp} (r \times s) [X]p))$
$= \prod_{A1,A2,\ldots,An,B1,B2,\ldots,Bm} (\sigma_{s.Bj = cs \wedge r.Ai = cr} (\sigma_{p.Ck = cp} (r \times s))$
$[X] p)$
$= \prod_{A1,A2,\ldots,An,B1,B2,\ldots,Bm} (\sigma_{s.Bj = cs \wedge r.Ai = cr \wedge p.Ck = cp} (r \times s [X]$
$p))$

Join antara relasi r dan s pada query SQL ini adalah *cartesian product.*

Tidak Berbasiskan Join

<u>Gabungan (*Union*)</u>

Operator *Union* digunakan terhadap relasi yang sama aritasnya sehingga semua *tuple* hasil penggabungan memiliki komponen yang sama. $\prod_{atribut}(\sigma_{ekspresi}(R)) \cup \prod_{atribut}(\sigma_{ekspresi}(S))$ dengan **R** dan **S** adalah relasi yang berkepentingan.

Bentuk Umum Sintaks Query SQL dengan menerapkan operator ini adalah sebagai berikut:

```
(Select {<relasi.atribut>}^n_1
From {<nama-relasi>}^n_1
[Where <eksp>] ) union
(Select {<relasi.atribut>}^n_1
      From {<nama-relasi>}^n_1
      [Where <eksp>] )
```

Selisih (*Set-difference*)

Operator *Set-difference* digunakan untuk mencari *tuple* yang terdapat pada satu relasi tetapi tidak terdapat pada relasi yang lain.

$$\Pi_{atribut}(\sigma_{ekspresi}(R))-\Pi_{atribut}(\sigma_{ekspresi}(S))$$

dengan **R** dan **S** adalah relasi yang berkepentingan.

Irisan (*Intersection*)

Operator *Intersection* digunakan untuk mencari *tuple* yang terdapat pada beberapa relasi.

$$\Pi_{atribut}(\sigma_{ekspresi}(R))\cap\Pi_{atribut}(\sigma_{ekspresi}(S))$$

dengan **R** dan **S** adalah relasi yang berkepentingan.

Pembagi (*Quotient / division*)

Sintaks Query SQL dengan menerapkan operator *quotient/division* tidak terdefinisi pada PQL karena tidak berbasiskan pada **natural join** dan merupakan komposisi dari operator *Set-difference* dan *cartesian product*.

$$\Pi_{atribut\ R\text{-}S}(R)-\Pi_{atribut\ R\text{-}S}((\Pi_{atribut\ R\text{-}S}(R)x\ S)-R)$$

dengan **R** dan **S** adalah relasi yang berkepentingan.

Join Selain Natural Join

Tidak Bersarang

Semua Sintaks Query SQL tidak bersarang melibatkan 1 relasi dapat dikonversi menjadi Sintaks Query PQL, sedangkan yang melibatkan n relasi hanya berbasiskan natural join yang dapat dikonversi.

Select {<relasi.atribut>}$^n_1$
From {<nama-relasi>}$^n_2$
[***Where*** <eksp>];
<eksp> ::= <eksp1> | **not (**<join-list-atribut-sama>**)**

Penjelasan:
Adanya **not** di depan <join list atribut sama> menghalangi terbentuknya natural join.

Berikut contoh kasus SQL tidak bersarang melibatkan n relasi tidak terdefinisi pada PQL:

Kasus 7
Misal terdapat relasi:
r (*A1, A2,…, An*)
s (*A1,B1,…, Bm*)

Sintaks Query SQL:
Select r.A_1, A_2,…, A_n, B_1,…, B_m
From r, s
Wherenot(r.A_2=cr and r.A_1=s.A_1)

Sintaks Query **PQL:** tidak terdefinisi

Bersarang

1 (satu) Relasi

Bentuk umum Sintaks Query SQL berikut tidak terdefinisi pada PQL karena menggunakan operator lojik **or** pada atribut yang seharusnya menjadi join antara relasi dan **not** di depan Sintaks Query join yang menyebabkan bukan natural join.

Select {<atribut>}$^n_1$
From <nama-relasi>
Where <nesteksp>
<nesteksp> ::= <nesteksp1>
|(<nesteksp1>)<oplojbin>(<nesteksp1>)
<nesteksp1> ::= <eksp> <**or**> <atribut>
<in>(<qeksp>) |<eksp> <**and**> <atribut> <**not**>
<in>(<qeksp>)
<qeksp>::=<***select atribut***> <***from***> <nama-relasi>
[<***where***><eksp>|
<nesteksp>]

Berikut contoh kasus SQL bersarang melibatkan 1 relasi tidak terdefinisi pada PQL:

Kasus 8
Misal terdapat relasi:
r (*A1, A2,…, An*)
s (*A1,B1,…, Bm*)

Sintaks Query SQL:
Select A_1, A_2,…, A_n
From r
Where A_2=cr or A_1 in
(***select*** A_1 ***From*** s)

Sintaks Query PQL: tidak terdefinisi

n Relasi

Bentuk umum Sintaks Query SQL bersarang melibatkan n relasi sebagaimana dijelaskan pada tabel berikut, tidak terdefinisi pada PQL.

Penjelasan ekivalensi dengan Sintaks Query PQL: bentuk umum ini tidak terdefinisi pada PQL karena adanya or dan not di depan Sintaks Query join untuk membentuk natural join.

Berikut contoh kasus Sintaks Query SQL bersarang melibatkan n relasi tidak terdefinisi pada PQL:

Select {<relasi.atribut>}$^{n}_{1}$
From {<nama-relasi>}$^{n}_{2}$
Where <nesteksp>
<nesteksp> ::= <nesteksp1>|
(<nesteksp1>)<oplojbin>(<nesteksp1>)
<nesteksp1> ::= <eksp><**and**><join list-atribut sama><**or**> <atribut> <in>(<qeksp>)| <eksp><**or**>< join list-atribut sama> <**and**><atribut><in> <qeksp>)|
<eksp><**or**>< join list-atribut sama ><**or**>
<atribut><in> <qeksp>| <eksp><**and**><**not**>< join list-atribut sama > <**and**><atribut>
<**not**><in><qeksp> | **not (**<eksp><**and**>
< join list-atribut sama ><**or**> <atribut><in><qeksp>**)**
<qeksp> ::= **<select atribut> <from>** <nama-relasi>
[**<where>**<eksp>|<nesteksp>]

Kasus 9
Misal terdapat relasi :

r (*A1, A2,…, An*)
s (*A1,B1,…, Bm*)
p(*A1,C1,…, Co*)

Sintaks Query **SQL :**
Select r.A_1, A_2,…, A_n, B_1,…, B_m
From r, s
Where r.A_2 = cr and r.A_1 = s.A_1 or s.A_1 in (select A_1 from p)

Sintaks Query **PQL:** tidak terdefinisi

CAKUPAN KONVERSI

Dari uraian di atas dapat diambil kesimpulan bahwa tidak semua Sintaks Query SQL dapat dikonversi menjadi Sintaks Query PQL Hanya Sintaks Query SQL yang berbasiskan natural join yang dapat dikonversi menjadi Sintaks Query PQL. Hal ini disebabkan karena PQL menerapkan teknik penggabungan pada algoritma jaringan semantik data (operator komposisi) [4], yang didasarkan pada prinsip operator join untuk relasi yang berkepentingan melalui atribut kunci (*virtual flat relation*) dan navigasi, yang didasarkan pada prinsip **natural join.**

SQL Terdefinisi pada SQL

Dari keseluruhan uji coba konversi SQL menjadi PQL yang telah dijelaskan, maka tabel berikut ini merupakan rangkuman bentuk umum sintaks SQL yang terdefinisi pada PQL.

Select {<relasi.atribut>}$^{n}_{1}$
From {<nama-relasi>}$^{n}_{1}$;
[***Where*** <nesteksp>] ;
<relasi.atribut>::= <relasi.atribut> | <atribut>
<atribut> ::= <huruf> {<huruf> |<angka>| _ }$^{n}_{0;}$
<nama-relasi> ::= <huruf>{<huruf> |<angka>| _ }$^{n}_{0}$
<nesteksp>::= <eksp> |<nesteksp1>| (<nesteksp1>)
<oplojbin>(<nesteksp1>);
<nesteksp1> ::= [<eksp> **<and>**] <join list-atribut sama> | <eksp><**and**>] <atribut><in>(<qeksp>)
|[<eksp><**and**>] <join list-atribut sama><**and**>
[<relasi.>] <atribut><in>(<qeksp>) ;
<qeksp> ::= **<select atribut> <from>** <nama-relasi>
[**<where>** <eksp>| <nesteksp>]
join list-atribut sama>::= {<relasi.atribut>
<relasi.atribut>[<**and**>]}$^{n}_{1}$
<oplojbin> ::= and | or
<eksp> ::= <eksp1>| (<eksp1>)
{<oplojbin><eksp>}$^{n}_{0}$| **not** (<eksp1>)
{<oplojbin><eksp>}$^{n}_{0}$
<eksp1> ::= [<relasi.>]
<atribut><oprel>[<relasi.>]<atribut>|
[<relasi.>]<atribut><oprel><konstanta>
<oprel> ::= <>|=|<|>|<=|>=
<konstanta> ::= <string>|<bilangan>
<string> ::= {<huruf> | <angka> | <spesial>}$^{n}_{0}$
<bilangan> ::= [-|+]{<bulat> [<bulat>]}
<bulat> ::= {<angka>}$^{n}_{1}$
<angka> ::= 0|1|2|3|4|5|6|7|8|9
<huruf> ::= a|b|c|d|…|z|A|B|C|D|…|Z
<spesial> ::= / | \ |*|&|(|) | ^ | % | $ | # | @ | ! | ? |
. | , | ” | ’ | [|] | { | } | | | ` | ~ | : |
; | + | - | _

Penjelasan*:*

- Secara umum, untuk sintaks Query SQL melibatkan n relasi pada where harus ada pemeriksaan keanggotaan yang sama pada atribut join agar membentuk **natural join** sehingga Sintaks Query PQL terdefinisi**.**
- Untuk atribut yang hanya ada pada satu relasi tidak harus disebutkan relasi-nya pada kondisi where, tetapi bila relasi pada from lebih dari satu maka akan lebih jelas bila relasi diikutsertakan.

SQL Tidak Terdefinisi pada PQL

Sedangkan cakupan SQL yang tidak terdefinisi pada PQL, dirangkum pada Tabel 6.

Tabel 6. Sintaks SQL Tidak Terdefinisi

***Select** {<relasi.atribut>}$^{n}_{1}$*
From [<nama-relasi>}$^{n}_{1}$
[***Where*** <nesteksp>][**)**]
[<oprtNonJoin>
(*Select* {<relasi.atribut>}$^{n}_{1}$
From [<nama-relasi>}$^{n}_{1}$
[***Where*** <nesteksp>]**)**]

<nesteksp> ::= <nesteksp1>
|(<nesteksp1>)<oplojbin>(<nesteksp1>);
<nesteksp1> ::= <eksp>[<**and**>< join list-atribut sama >] <**or**>[<relasi.>] <atribut><in>(<qeksp>) | <eksp><**or**> <join list-atribut sama > <**and**> [<relasi.>]<atribut><in>(<qeksp>)| <eksp><**or**>< join list-atribut sama> <**or**> [<relasi.>] <atribut><in><qeksp>| <eksp> <**and**><**not**>< join list-atribut sama > <**and**>[<relasi.>] <atribut> <**not**><in><qeksp> | **not(**<eksp><**and**> <join list-atribut sama ><**or**> [<relasi.>]<atribut><in><qeksp>**)** | **not (**<join list-atribut sama>**) ;**
<qeksp>::= ***<select atribut>*** ***<from>*** <nama-relasi>[***<where>*** <eksp> |<nesteksp>];
<oprtNonjoin> ::= **union |intersect | except**

Penjelasan*:*

- Semua Sintaks Query SQL yang tidak berbasiskan natural join atau yang menghalangi natural join dengan menambahkan **not** di depan Sintaks Query join atribut serta pemeriksaan keanggotaan sama hanya pada sebagian atribut join**,** tidak dapat dikonversi menjadi Sintaks Query PQL.
- Sintaks Query SQL yang berbasiskan *cartesian product* dan yang tidak berbasiskan join, tidak dapat dikonversi menjadi Sintaks Query PQL.

KESIMPULAN

Konversi sintaks SQL menjadi sintaks PQL dapat dilakukan, terbatas untuk query SQL yang berbasiskan natural join, yang melibatkan 1 atau beberapa relasi, baik yang bersarang (nested) maupun yang tidak bersarang.

Sebagai dampak dari karakteristik hasil eksekusi PQL, maka penggunaan fungsi agregat seperti GROUP BY, DISTINCT, UNIQUE, tidak perlu dikonversi, karena pengertian dari fungsi tersebut sudah secara eksplisit menjadi bagian dari konsep PQL, khususnya yang dijabarkan pada hasil eksekusi PQL, yang disebut sebagai Composite Tupple [4].

DAFTAR PUSTAKA

1. Atzeni, Paolo dan Valeri de Antonellis, *Relational Database* Theory, Redwood City, 1993
2. *www.microsoft*.com/technet/prodtechnol/sql/2000/reskit/part9.
3. Sitohang Benhard, *Bahasa Penelusuran Basis Data PQL*, Lab. Basis Data, Jurusan Teknik Informatika, ITB, 1995
4. Sitohang Benhard, *PQL: Conversion of Syntax to SQL, World Multiconference on Systemics, Cybernetics and Informatics*, SCI 2001, Orlando, USA, Juli 2001.
5. Sitohang Benhard, *PQL: Operasi komposisi dan jaringan semantik data*, Proceedings ITB, ISSN 0125-9350 Vol. 33, No. 2, 2001.
6. Sitohang Benhard, *Public Query Language/PQL*, Conference on Electrical, Electronics, Communication and Information CECI 2001, 7-8 Maret 2001, Indonesia
7. Ullman Jeffrey D., *Principles of Database Systems*, 1988.
8. Winter Alfred, Haux Reinhold, *A Modified RM/T Data Model for a Universal Relation View*, Dept. of Medical Informatics, Univ. of Heidelberg, Germany, 1994.
9. Yuni Widiastuti, *Algoritma Konversi Sintaks Query PQL Menjadi Sintaks Query SQL*, Tugas Akhir, Jurusan Teknik Informatika, ITB, 2001.
10. Firdaus Zulfahmi, *Eksekusi PQL menggunakan DBMS-SQL: Algoritma Konversi Sintaks Query PQL menjadi Sintaks Query SQL*, Tugas Akhir, Jurusan Teknik Informatika, ITB, 2000.